Extending homotopy type theory with strict equality

Altenkirch, Thorsten; Capriotti, Paolo; Kraus, Nicolai

DOI:
10.4230/LIPIcs.CSL.2016.21

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Altenkirch, T, Capriotti, P & Kraus, N 2016, Extending homotopy type theory with strict equality. in J-M Talbot & L Regnier (eds), *25th EACSL Annual Conference on Computer Science Logic 2016 (CSL 2016).*, 21, Leibniz International Proceedings in Informatics, LIPIcs, vol. 62, Schloss Dagstuhl, pp. 21:1-21:17, 25th EACSL Annual Conference on Computer Science Logic, CSL 2016 and the 30th Workshop on Computer Science Logic, Marseille, France, 29/08/16. https://doi.org/10.4230/LIPIcs.CSL.2016.21

Extending Homotopy Type Theory with Strict Equality*

Thorsten Altenkirch[1], Paolo Capriotti[2], and Nicolai Kraus[3]

1 **University of Nottingham, School of Computer Science, Nottingham, UK**
`thorsten.altenkirch@nottingham.ac.uk`
2 **University of Nottingham, School of Computer Science, Nottingham, UK**
`paolo.capriotti@nottingham.ac.uk`
3 **University of Nottingham, School of Computer Science, Nottingham, UK**
`nicolai.kraus@nottingham.ac.uk`

Abstract

In homotopy type theory (HoTT), all constructions are necessarily stable under homotopy equivalence. This has shortcomings: for example, it is believed that it is impossible to define a type of semi-simplicial types. More generally, it is difficult and often impossible to handle towers of coherences. To address this, we propose a 2-level theory which features both strict and weak equality. This can essentially be represented as *two* type theories: an "outer" one, containing a strict equality type former, and an "inner" one, which is some version of HoTT. Our type theory is inspired by Voevodsky's suggestion of a *homotopy type system* (HTS) which currently refers to a range of ideas. A core insight of our proposal is that we do not need any form of equality reflection in order to achieve what HTS was suggested for. Instead, having unique identity proofs in the outer type theory is sufficient, and it also has the meta-theoretical advantage of not breaking decidability of type checking. The inner theory can be an easily justifiable extensions of HoTT, allowing the construction of "infinite structures" which are considered impossible in plain HoTT. Alternatively, we can set the inner theory to be exactly the current standard formulation of HoTT, in which case our system can be thought of as a type-theoretic framework for working with "schematic" definitions in HoTT. As demonstrations, we define semi-simplicial types and formalise constructions of Reedy fibrant diagrams.

1998 ACM Subject Classification F.4.1 Mathematical Logic: Lambda calculus and related systems

Keywords and phrases homotopy type theory, coherences, strict equality, homotopy type system

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.21

1 Introduction: Motivations for a 2-Level System

The identity type is probably the single concept of intensional Martin-Löf type theory (MLTT) which has created most questions, stimulated most research, caused most confusion, and enabled the largest number of different views. Written $\mathsf{Id}_A(x, y)$ for elements x, y of a type A, the identity type expresses that two elements are equal in some sense and can be substituted for each other, and the elements of this type are called equalities. However, by default, it has a somewhat strange standing. On the one hand, it is not well-behaved when it comes to describing equality of functions and equality of types. Given two functions of the same type,

* This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), grant reference EP/M016994/1, and by USAF, Airforce office for scientific research, award FA9550-16-1-0029.

we cannot derive the principle of (naive) function extensionality, saying that the functions are equal if they are equal at every point, from the basic axioms. We also cannot show that equivalent types can be substituted for each other, although they do behave equivalently in any given situation. On the other hand, we also cannot derive the principle of unique identity proofs (UIP): by a construction of Hofmann and Streicher [11], we can not show $\mathsf{Id}_{\mathsf{Id}(x,y)}(p,q)$ for two equalities p, q. A priori, it is unclear what it should mean to have distinguishable equalities and how one can make sense of this behaviour.

As we view it, there are two major ways to remedy the situation. Both can be seen as extensions of MLTT. The principle of function extensionality can be added in any case, but after that, we have two possibilities to extend the theory further. The first is to add UIP (or, equivalently, Streicher's K) as an axiom. Let us call the resulting type theory MLTT_K. The second possibility is to consider univalence, ensuring that type equality is what one would ideally expect. This approach is taken by Homotopy Type Theory [21], and we write HoTT for the resulting theory.

One appeal of HoTT is that equalities can be seen as paths in a space, and it is even possibly to develop a lot of homotopy theory synthetically. An important insight is that, when doing homotopy theory in type theory, every statement that we make is up to homotopy, and every construction respects (homotopical) equivalence. This means that whatever we do will be "non-evil" in the sense that it can only take the homotopy type of spaces, and homotopy equivalence classes of maps, into account, and not the concrete representations of spaces or maps. Exactly this is often considered a selling point of HoTT: one often defines constructions using representatives of homotopy classes in traditional homotopy theory and is forced to show that the constructions are well-defined, i.e. do not depend on the choice of the representative. In HoTT, everything we do is automatically well-defined as we are simply not able to talk about strict properties internally.

Going back to the homotopical point of view, it is not hard to imagine that the blessing of having only constructions up to homotopy can turn out to be a curse: we are unable to make any strict statement. For example, we cannot form a type expressing that a given diagram commutes strictly; all we can do it stating that it commutes up to homotopy. Unfortunately, depending on the shape of the diagram, this will only be sufficient in the simplest cases. More often than not, it will be necessary to say that the different "pieces" (the equalities expressing commutativity) fit together. For example, the fact that a certain sub-diagram commutes can be part of the proof that the diagram commutes, but it may at the same time be derivable as the composition of the fact that other sub-diagrams commute. In this case, it is natural to require these different ways of getting a certain proof to be equal. It does not stop here; these new proofs can itself be required to be coherent, and so on. What happens here is not at all something that can only be observed in type theory. The first step becomes already apparent in the theory of monoidal categories in the form of "Mac Lane's Pengaton". On higher dimensions, it is exactly the same issue that is discussed as *homotopy commutativity versus homotopy coherence* by Lurie [16].

In general, homotopy coherence corresponds to infinite towers of coherence data, and it is a major open problem (and commonly believed to be unsolvable) to express such towers internally in HoTT. One way to avoid this problem is to restrict constructions to types of low truncation levels. As an examples, the category theory developed in [1] only considers 1-truncated types and [what corresponds to] ordinary categories. This is in many situations not satisfactory: we know that types are ∞-groupoids [15, 22], and similarly, the universe should be an $(\infty, 1)$-category. Unfortunately, we seem to have no way of expressing this internally in HoTT.

The crucial shortcoming of HoTT is that we are unable to perform some constructions which actually *seem* to be harmless as they only require finite amounts of coherences at every step. An example that has received considerable attention in the HoTT-community is the construction of Reedy fibrant n-truncated semi-simplicial types (simply referred to as *semi-simplicial types*). Let us start with Δ_+, the category of finite non-zero ordinals and strictly monotonous functions. Let us write $[\mathsf{n}]$ for the ordinal with $(n+1)$ elements. A type-valued diagram over Δ_+^{op} is a strict functor from Δ_+^{op} to the category of types. It would correspond to a type $X_{[\mathsf{n}]}$ (for simplicity written X_n) for every n, and face maps $d_i : X_{n+1} \to X_n$ for $0 \leq i \leq n$, as it is well-known that any map can be written as a composition of face maps. The problem is that we need the semi-simplicial identities (essentially a representation of the functor laws) to be strict, which we cannot express in type theory. The considered approach to avoid this problem is to only attempt internalising Reedy fibrant diagrams over Δ_+^{op}, essentially ensuring that the face maps are simple projections. Using the correspondence between fibrations and type families, a (Reedy fibrant) semi-simplicial type then corresponds to a type X_0 (the "points") on level 0. On level 1, we need a family $X_1 : X_0 \to X_0 \to \mathcal{U}$, where $\mathcal{U}$ is the universe of types. We think of X_1 as lines between types. Next, we need $X_2 : \Pi_{a,b,c:X_0} X_1(a,b) \to X_1(b,c) \to X_1(a,c) \to \mathcal{U}$, the type of fillers for triangles. Writing down the type of X_4 is already a bit tedious, but nevertheless straightforward: X_4 is a family which gives a type for any collection of four points, six lines and four triangles that form an empty tetrahedron. The long-standing open problem of homotopy type theory is to write down the type of X_n in general (up to equivalence). Perhaps surprisingly, this does not seem to be possible. What *is* possible is to generate an expression X_n for every externally fixed numeral n, such that the expressions $X_0, X_1, X_2, \ldots$ all "fit together". When one tries to do the construction for a *variable* $n : \mathbb{N}$, it does no longer type-check. The reason is that some judgmental equalities that hold in the case of a numeral n fail to hold in the case of a variable. We can try to prove in type theory that the required equalities hold up to homotopy. However, we quickly have to realize that we then also need that these equalities are coherent, and that the coherence proofs are coherent themselves, and so on; something that no one has managed to do so far. The problem is that we cannot formulate the tower of coherences that we need to prove. Morally, the required equalities *should* hold and be fully coherent just because they are trivially satisfied for each externally fixed natural number. If we can use a system where we judgmental equalities can be shown by induction, there would thus be no problem at all; however, this would require judgmental equalities to be some sort of type.

In MLTT_K, the internal equality type can be seen as an internalised version of judgmental equality. For example, a well-known meta-theoretic statement is that any equality that is constructed in the empty context is refl; that is, if we can show an equality internally without assumptions, then this equality holds judgmentally. Not surprisingly, it is possible to construct Reedy fibrant semi-simplicial types in MLTT_K. However, we can also simply define categories and functors in the naive sense, as all coherences are satisfied automatically.

The idea of a 2-level system is to combine MLTT_K and HoTT instead of viewing them as two alternative extensions of MLTT. We can describe this in two ways:

1. Start with a type theory that has axiom K and consider a "sub-theory" of types and maps that do not talk about equalities. Inside this sub-system, we can consider a new equality type and univalent universes. If we use the equality type with K of the outer system, we cannot form types that live in the inner system; however, we can reason about the inner system.
2. We may start with HoTT and try to formulate the meta-theory (in which judgmental equality lives) as a type system. It is not necessary to capture every aspect of the meta-theory in this type system; the important part is that this outer type system has

an equality type (which we call *strict equality*) satisfying K. We then have in total three equalities: the equality in HoTT; the strict equality; and the judgmental equality (which we should now refer to as *definitional equality*). From the point of view of HoTT, the strict equality and definitional equality are identical.

Considering a type theory with two equality types is not new. Our proposal is motivated by the suggestion of a *homotopy type system* (HTS) by Voevodsky [23]. However, as far as we are aware, "HTS" mostly refers to a range of ideas so far but not to a precise theory, and there is no publication that presents or analyses HTS. The core idea of HTS is to make some judgmental equalities provable. In other words, some form of equality reflection, the characteristic concept of extensional type theory, is reintroduced, in a way that is compatible with the "standard" intensional identity type. A concrete theory that could be called "HTS" is outlined in the draft [23] which, unfortunately, presents rather involved rules that would presumably be non-trivial to justify. One original goal of *Andromeda*, a project by Bauer et al. [4], was to serve as an implementation of HTS. This is currently not the case as a fibrant fragment with a univalent universe has not yet been implemented, but future developments might go into that direction.

A key observation of the current paper is that no form of equality reflection is actually required. Our proposal instead only required unique identity proofs for the strict equality type. Thus, we can avoid all the problems that are usually connected to equality reflection, such as undecidability of type checking. In contrast, the theory that we suggest is well-behaved, very close to the standard formulation of HoTT, and has straightforward semantics. One could expect that a downside of our system might be reduced expressibility compared to a theory that features equality reflection. However, we show that we can achieve in our system what HTS was suggested for: a definition of semi-simplicial types, and other constructions. This should actually not be surprising in the light of Hofmann's result [9], which states that equality reflection is conservative over MLTT_K.

Our 2-level theory can be defined as two separate type theories with a morphism between them. This actually gives a recipe for constructing a variety of reasonable 2-level theories, and the choices that can be made affect the exact abilities of the system. We believe that our 2-level theories can be used in two ways. First, we can use the outer theory as a powerful formal language to study the inner theory. For some formulations of the 2-level theory, we get a conservativity result (by an argument of the second-named author; see the forthcoming thesis [5]). This means that the inner theory is exactly HoTT as studied in the standard textbook on homotopy type theory [21] and by many authors. In a proof assistant which supports this theory, we can then implement results that so far can only be stated meta-theoretically. To give an example, it is shown in [13] that constant functions from A to B which satisfy n coherence conditions correspond to maps $\|A\|_{-1} \to B$, provided that B is n-truncated. This can be done in HoTT only if n is an externally fixed natural number. In the 2-level system, we can formalise it by taking n to be a number in the outer theory, and show that the equivalence holds in the inner theory.

Second, we can use the construction of 2-level theories to derive extensions of HoTT that allow constructions that HoTT does not allow. For example, we can assume that the natural numbers of the outer theory are *exactly* the natural numbers of the inner theory, something that is satisfied in the simplicial set model. This gives us a univalent type theory in which various concepts including semi-simplicial types can be defined.

Contributions of the paper. Summarised, the main contributions of the paper are:

- We give (for the first time) a clean presentation of a system with two equalities.

- Our theory is simple enough to have straightforward semantics. Such semantics have not yet been described for previous proposals [23, 18].
- We demonstrate how our theory allows constructions that are thought to be impossible in standard HoTT, such as semi-simplicial types [8, 18]. A partial Agda formalisation is available.
- Schematic constructions, which could so far only be given on paper, can be formalised in our system. As an example, we perform constructions of Reedy fibrant diagrams. Further work is outlined in the conclusions.

Related work. The current paper is the write-up of our presentation [3] at TYPES'15. As briefly explained above, the main difference to Voevosky's draft [23] is that we do not consider any form of equality reflection, saving us from various difficulties.

Superficially related is the construction by Maietti [17] of a *two-level foundation for constructive mathematics*. However, their motivation and goals are very different from ours, hence their system cannot be used to reconcile strict equality with univalence.

Our work is more closely related to a recent proposal by Part and Luo [18] of Logic-enriched HoTT. In their system, our strict layer of type theory is replaced by a "logic enrichment". Their proposal is limited to the construction of semi-simplicial types (corresponding to the one that we give in Section 3). It is not explained whether this can be generalised to Reedy fibrant diagrams in the sense we present in Section 4, as they use specific properties of the Δ_+ category.

Herbelin has given a construction of semi-simplicial types along the lines of the one in Section 3 in an unspecified type theory containing a "connective" for strict equality [8].

Agda formalisation. As a proof assistant based on our 2-level theory does not (yet) exist, we cannot formalise our constructions exactly as they are presented. However, we have implemented in Agda an approximation of the construction of semi-simplicial types that is given in Section 3. It can be found on GitHub at `github.com/nicolaikraus/HoTT-Agda/tree/master/nicolai/SemiSimp`. For an explanation of the relationship between the construction given in the paper and this implementation, we refer to the last remark of Section 3.

Organisation. The structure of the paper is as follows. In Section 2, we specify our 2-level theories. Section 3 explains the construction of Reedy fibrant n-truncated semi-simplicial types in a way that could *nearly* be done in homotopy type theory, and we show how the missing gap is filled by our strict equality. Then, in Section 4, we demonstrate how our theory can be used to internalise standard constructions in a fairly straightforward way. Finally, in Section 5, we outline further work and conclude the paper.

2 The Specification of a 2-Level System

In this section, we want to specify our 2-level theory (or, to be precise, our family of 2-level theories). We give two presentations: first, the semantical approach, and second, the syntactical approach. With the first approach, we explain how the theory is constructed. It also shows which choices can be made, and how models of the 2-level theory can be constructed. The syntax that we propose afterwards is based on the semantics, but fixes a precise system.

2.1 Semantical Approach

Many models of type theory consist of a category $\mathcal{C}$, modelling the category of contexts. Starting from $\mathcal{C}$, additional structures are added to model types and terms, together with the structure that is needed to model the components of the considered theory (such as universes or dependent functions). Then, a model of a 2-level theory in our sense is given by a category of contexts $\mathcal{C}$, together with *two* structures on $\mathcal{C}$ such that the first structure (taken together with $\mathcal{C}$) models HoTT, and the second structure (taken together with $\mathcal{C}$) models MLTT_K. Finally, we need a morphism between the structures in a suitable sense, describing how any type or term in HoTT can be viewed as a type or term in MLTT_K.

We make this precise using the notion of *categories with families* [6]. Let us recall the definition:

▶ **Definition 1** (CwF [6])**.** A *category with families* (CwF) is given by:
- a category $\mathcal{C}$, equipped with a distinguished terminal object 1;
- a presheaf $\mathsf{Ty} : \mathcal{C} \to \mathsf{Set}^{\mathrm{op}}$;
- a presheaf $\mathsf{Tm} : \int \mathsf{Ty} \to \mathsf{Set}^{\mathrm{op}}$;
- for all $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, an object $(\Gamma.A, \pi_A) : \mathcal{C}/\Gamma$ representing the functor $\mathcal{C}/\Gamma \to \mathsf{Set}^{\mathrm{op}}$ defined by:

$$(\Delta, \sigma) \mapsto \mathsf{Tm}_\Delta(A[\sigma]).$$

The objects of $\mathcal{C}$ are called *contexts*. Given a context Γ, the elements of $\mathsf{Ty}(\Gamma)$ are called *types*, and given a type A, the elements of $\mathsf{Tm}_\Gamma(A)$ are called *terms*.

The context $\Gamma.A$ is called the *context extension* of Γ by the type A, and π_A is the *display map* of A.

The action of Ty and Tm on morphisms is called *substitution*.

A CwF can be regarded as a model of MLTT with only *structural rules*, i.e. rules that deal with types, terms and substitutions, but no type formers (like Π or Σ types). Type formers can be postulated separately as additional structures on a CwF. For details, we refer to [6] and [10].

To model a 2-level type theory, we need to add some extra structure to a CwF:

▶ **Definition 2.** A *2-level category with families* is a CwF $\mathcal{C}$, together with:
- a presheaf $\mathsf{Ty}^{\mathrm{f}} : \mathcal{C} \to \mathsf{Set}^{\mathrm{op}}$;
- a natural transformation $|-| : \mathsf{Ty}^{\mathrm{f}} \to \mathsf{Ty}$.

Given a 2-level CwF $\mathcal{C}$, we will denote the underlying category with family by $\mathcal{C}^{\mathrm{s}}$. There is also a second CwF structure on $\mathcal{C}$, where the types are given by Ty^{f}, and terms are defined as:

$$\mathsf{Tm}^{\mathrm{f}}_\Gamma(A) = \mathsf{Tm}_\Gamma(|A|),$$

and context extension is given simply by $\Gamma.A = \Gamma.|A|$. We will denote this second CwF by $\mathcal{C}^{\mathrm{f}}$.

The map $|-|$ determines a morphism of CwF $\mathcal{C}^{\mathrm{f}} \to \mathcal{C}^{\mathrm{s}}$, which we will also denote by $|-|$.

The theory employed in this paper will be modelled by a 2-level CwF $\mathcal{C}$ where:
- C^{s} is a model of MLTT_K;
- C^{f} is a model of HoTT;
- the morphism $|-|$ preserves Π, Σ and 1 *strictly*.

Note that, crucially, equality types, although present in both CwF structures, are *not* generally preserved. This is important, because preservation of equality would mean that axiom K holds in $\mathcal{C}^{\mathrm{f}}$, which in turn would imply that $\mathcal{C}^{\mathrm{f}}$ does not admit any univalent universes containing non-propositional types.

Other type formers besides those mentioned might or might not be preserved. We say that a 2-level CwF is *strong* if $|-|$ preserves coproducts, natural numbers, and the empty type (more generally W-types, if part of the theory).

Interestingly, most of the existing models of HoTT can be naturally extended to a 2-level CwF. Most notably, the simplicial model [12] can be regarded as a 2-level CwF, where Ty is given by arbitrary (well-ordered) morphisms, Ty^{f} is the subfunctor of Ty consisting of Kan fibrations, and $|-|$ is simply the inclusion. With this setup, C^{s} is (equivalent to) a presheaf CwF, which models type theory with equality reflection (hence, in particular, MLTT_K), and C^{f} is the same as the model defined in the paper.

One can also start with an arbitrary model $\mathcal{C}$ of HoTT, then consider the presheaf category $\widehat{\mathcal{C}}$. It is perhaps not surprising that one can equip $\widehat{\mathcal{C}}$ with a 2-level CwF structure so that $\mathcal{C}$ can be recovered inside $\widehat{\mathcal{C}}^{\mathrm{f}}$. This makes it possible to use 2-level type theory to formulate and prove statements that hold in any model of HoTT, i.e. 2-level type theory can be regarded as a meta-language for HoTT.

However, the details of this construction are somewhat involved, mainly due to the strictness requirement in Definition 2. Therefore, we will not explore that direction further in this paper and refer instead to the forthcoming thesis of the second-named author [5].

2.2 Syntactical Approach

In the syntactical approach, the clear separation of a 2-level theory into two theories becomes harder to see. We do not leave as many choices open as in the semantical approach, but rather fix a concrete theory; and the choices that we make ensure that the conservativity result of the forthcoming thesis [5] applies to the presented theory.

For a precise specification, we choose a presentation in the style of [21, Appendix A.2], which considers three forms of judgments: $\Gamma \vdash \mathrm{ctx}$; $\Gamma \vdash a : A$; and $\Gamma \vdash a \equiv a' : A$. Fortunately, we do not need to give *all* the rules, as most of them are identical to those given in [21, Appendix A.2]. Thus, in most cases, it is sufficient to state the difference in order to give both an understandable and a precise specification.

The theory that we consider has the following basic types and type formers: Π, the type former of dependent functions; Σ, the type former of dependent pairs; $+$, the coproduct type former; $\mathbf{1}$, the unit type; $\mathbf{0}$, the empty type; $\mathbb{N}$, the fibrant type of natural numbers; $=$, the equality type (in the sense of HoTT); a hierarchy $\mathcal{U}_0, \mathcal{U}_1, \ldots$ of universes. So far, we can think of these as the types and type formers of HoTT. Further, we have: $+^s$, the strict coproduct; $\mathbf{0}^s$, the strict empty pretype; $\mathbb{N}^s$, the strict pretype of natural numbers; $\stackrel{s}{=}$, the strict equality; and hierarchy $\mathcal{U}_0^s, \mathcal{U}_1^s, \ldots$ of strict universes.

Both the hierarchy $\mathcal{U}_0, \mathcal{U}_1, \ldots$ and the hierarchy $\mathcal{U}_0^s, \mathcal{U}_1^s, \ldots$ are cumulative. We think of the elements of $\mathcal{U}_i$ as *fibrant types* (or simply *types*), while the elements of $\mathcal{U}_i^s$ are *pretypes*.

Recall possibility 1 from the two ways of describing a 2-level system as outlined on page 3: we can start with a type theory with K and embed HoTT later. Thus, we first consider the type theory with the basic types $\mathbf{0}^s$, $\mathbf{1}$, $\mathbb{N}^s$, with universes $\mathcal{U}_0^s, \mathcal{U}_1^s, \ldots$, and with $+^s$, Π, and Σ. All rules correspond exactly to those of [21, Appendix A.2]. For example:

- Contexts are formed using elements of $\mathcal{U}_i^s$, i.e. if Γ is a context and $\Gamma \vdash A : \mathcal{U}_i^s$, then $\Gamma.A$ is a context.
- If $\Gamma \vdash A : \mathcal{U}_i^s$ and $\Gamma.A \vdash B : \mathcal{U}_i^s$, then we have $\Gamma \vdash \Pi_A B : \mathcal{U}_i^s$.

- If $\Gamma, x : A \vdash b : B$, then we have $\Gamma \vdash \lambda x.b : \Pi_A B$.
- All further rules of Π, and all rules of Σ, $+^s$, $\mathbf{0}^s$, $\mathbf{1}$, and $\mathbb{N}^s$ are also those given in [21, A.2.4–9]. The constructors of $+^s$ are called inl^s, inr^s, and the constructors of $\mathbb{N}^s$ are called $\mathbf{0}^s$ and succ^s. We assume all the usual judgmental rules (including the judgmental η-rule for Σ).

Further, the theory has a strict identity pretype, written $\overset{s}{=}$: For any $\Gamma \vdash A : \mathcal{U}_i^s$ and $\Gamma \vdash a_1, a_2 : A$, we have $\Gamma \vdash a_1 \overset{s}{=} a_2 : \mathcal{U}_i^s$, with the introduction rule refl^s, the eliminator J^s, and the usual computation rule. For pretypes $A, B : \mathcal{U}_i^s$, we can form the pretype of strict isomorphisms, written $A \simeq^s B$ (unlike in HoTT, it is enough to have maps in both directions such that both compositions are pointwise strictly equal to the identity). However, we do *not* assume that $\mathcal{U}_i^s$ is univalent. Instead, we add the rule K^s: for A, a_1, a_2 as before, and for $\Gamma \vdash p, q : a_1 \overset{s}{=} a_2$, we have a term $\Gamma \vdash K^s(p, q) : p \overset{s}{=} q$. We also assume that $\overset{s}{=}$ satisfies the principle of function extensionality.

Note that, so far, we have not considered $\mathcal{U}_i$, $+$, $\mathbf{0}$, $\mathbb{N}$, $=$ at all. We do this now, and their rules are more subtle. The first important rule is that any type (element of $\mathcal{U}_i$) is also a pretype (element of $\mathcal{U}_i^s$), as given by the inference rule (a) below. This means that informally we can understand $\mathcal{U}_i$ as a subtype of $\mathcal{U}_i^s$.

Now, let A and B be fibrant types, i.e. $\Gamma \vdash A : \mathcal{U}_i$ and $\Gamma.A \vdash B : \mathcal{U}_i$. Then, by (a) and by the formation rule of Π, we have $\Gamma \vdash \Pi_A B : \mathcal{U}_i^s$. However, we add the rule that, under these conditions, this conclusion can be lifted to $\Gamma \vdash \Pi_A B : \mathcal{U}_i$. In other words, Π preserves types. We add the same rule for Σ, as shown by the rule (b) below. We do *not* add the same rule for $+^s$, that is, the strict sum of two types is still only a pretype. Similarly, there is no special rule for $\overset{s}{=}$: if $\Gamma \vdash a_1, a_2 : A$, it does not matter whether A is a type or only a pretype, the expression $a_1 \overset{s}{=} a_2$ is only an element of $\mathcal{U}_i^s$, not of $\mathcal{U}_i$.

In contrast, the equality type former $=$ can only be applied to elements of fibrant types; i.e. its formation rule is given by the rule (c) below. Note that there is no strict universe $\mathcal{U}_i^s$ involved:

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_i^s} \quad (a) \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma.A \vdash B : \mathcal{U}_i}{\Gamma \vdash \Sigma_A B : \mathcal{U}_i} \quad (b) \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma \vdash a_1, a_2 : A}{\Gamma \vdash a_1 = a_2 : \mathcal{U}_i} \quad (c)$$

The type $a_1 = a_2$ (with the constructor refl) is a pretype by rule eq:type-is-pretype, but (usually) not the same as $a_1 \overset{s}{=} a_2$. The elimination principle of $=$ only works for families of types (not in general for pretypes). This means that the usual "path induction" principle, which allows us to construct an element of $\Pi_{a_1,a_2:A}\Pi_{p:a_1=a_2} P(a_1, a_2, p)$, can only be applied if P is a family of types, i.e. $\Gamma \vdash P : (\Sigma_{a_1,a_2:A} a_1 = a_2) \to \mathcal{U}_i$. If we restrict ourselves to types, we can do everything that we can do in HoTT. In particular, we can say what it means for a function between types to be an equivalences (using $=$). We assume that the universes $\mathcal{U}_0, \mathcal{U}_1, \ldots$ are univalent, that is, the canonical map from type of equalities $A = B$ to the type of equivalences $A \simeq B$ (defined as usual in homotopy type theory) is an equivalence itself.

Similarly, the type former $+$ only allows us to form a type $A + B$ if A and B are types (elements of some $\mathcal{U}_i$), and we can only defined a function $\Pi_{x:A+B} P(x)$ with the usual induction principle if P is a family of types.

We have the type of natural numbers $\mathbb{N} : \mathcal{U}_0$ (in any context) with the constructors $\mathbf{0}$, succ, and its induction principle can only be applied to eliminate into families of types. The same is the case for $\mathbf{0}$. This completes the syntactical characterization of our 2-level system. We will usually omit the index and simply write $\mathcal{U}^s$ or $\mathcal{U}$ instead of $\mathcal{U}_i^s$ or $\mathcal{U}_i$ in the same style as it is done in [21]. A strong 2-level theory is now simply one in which $\mathbf{0}^s$ and $\mathbf{0}$, and $+^s$ and $+$, and $\mathbb{N}^s$ and $\mathbb{N}$ coincide.

▶ Remark. If A is a ("fibrant") type with elements $a_1, a_2 : A$, then we can form both the type $a_1 = a_2$ and the pretype $a_1 \stackrel{s}{=} a_2$. By "strict path induction" (i.e. an application of J^s), we can easily construct a function $a_1 \stackrel{s}{=} a_2 \to a_1 = a_2$. Consequently, strictly equal elements of a type are also homotopy-equal. This corresponds to the fact that judgmental equality in HoTT implies equality ("refl"). We cannot construct a function in the other direction, as the path induction principle J can only be applied to eliminate into types, which $a_1 \stackrel{s}{=} a_2$ is not. Hence, equal elements are not necessarily strictly equal. However, if we have a type which does satisfy this "equality reflection" principle, it is easy to see that the type is a set in the sense of homotopy type theory.

3 Semi-Simplicial Types

In a 2-level theory, we can define strict categories in a reasonable sense. There are a number of choices that one can make; for example, the objects could be a fibrant type or only assumed to be a pretype. Later (see Definition 5), we will give one possible concrete definition. The important thing is that the categorical equations can be required strictly; and, if we have such a strict category $\mathcal{C}$, we can easily write down the pretype of strict functors $\mathcal{C} \to \mathcal{U}$. Unfortunately, there is no general way to get an actual fibrant type of such functors.

The case where $\mathcal{C}$ is Δ^{op} (the category of finite nonempty ordinals and increasing functions) is particularly interesting since "simplicial structures" appear frequently in homotopy theory. Having a type of functors $\Delta^{\mathrm{op}} \to \mathcal{U}$ would have many potential applications; maybe most notably, one could try to internalise a constructive version of the model of univalent foundations in simplicial sets [12]. Unfortunately, it seems unreasonable to expect that such a type can be constructed. It would be a good approximation (and potentially good enough for many constructions) if one could form a type of functors $\Delta_+^{\mathrm{op}} \to \mathcal{U}$, where Δ_+ is the category of finite nonempty ordinals and *strictly* increasing functions (a more precise definition will be given below). Trying to define such a type seems more promising, since Δ_+^{op} is an *inverse category* and, if we restrict ourselves to *Reedy fibrant* functors, we can describe them by induction (see [20]).

This gave rise to the challenge of defining Reedy fibrant n-truncated semi-simplicial types (in the community often just referred to as *semi-simplicial types*) in type theory. The challenge was first raised during the special year on Univalent Foundations at the Institute for Advanced Study (Princeton, 2012–13). As briefly sketched in the introduction, a (Reedy fibrant) 0-truncated semi-simplicial type is simply a type $X_0 : \mathcal{U}$, a 1-truncated semi-simplicial type is such an X_0 together with a family $X_1 : X_0 \to X_0 \to \mathcal{U}$, and so on. Defining n-truncated semi-simplicial types as a family $\mathsf{SST} : \mathbb{N} \to \mathcal{U}$ in homotopy type theory is a famous open problem. Many attempts (see e.g. [8, 19, 14]) have not led to a solution, and at a workshop on HoTT in Warsaw (June 29–30, 2015), a clear majority of the participants expected it to be impossible.

The hard part of the construction is to define the *matching objects* M_n, that is the "full boundary" of an n-simplex, as the corresponding component of SST_n is then just given as a family $M_n \to \mathcal{U}$. A popular attempt for defining the matching objects M_n is to define the *k-skeleton* SK_n^k of SST_n by induction on k, that is, the collection of components of SST_n up to level k. As long as k is a fixed numeral, this can be done. However, if k is a variable, some crucial judgmental equalities do not hold anymore and the construction is believed to become impossible. In our 2-level theory, we can prove strict equalities (i.e. the internalisation of judgmental equality) by induction. This allows us to complete the sketched approach of defining SST in a weak sense: we construct a family $\mathsf{SST} : \mathbb{N}^s \to \mathcal{U}$. If we assume that the

strict natural numbers ($\mathbb{N}^s$) and the fibrant ones ($\mathbb{N}$) coincide, this represents a construction of n-truncated semi-simplicial types. Without this assumption and under the conjectured conservativity result [5], it internalises the result that n-truncated semi-simplicial types can be defined for an externally fixed n.

To give the precise construction, let us first note that we have the family $\mathsf{Fin} : \mathbb{N}^s \to \mathcal{U}$ of finite types (Fin_n is the type with n elements), together with the families $<^n\colon \mathsf{Fin}_n \to \mathsf{Fin}_n \to \mathcal{U}$. Let us write $\mathsf{isIncr}_{i,j}$ for the predicate

$$\mathsf{isIncr}_{i,j} : (\mathsf{Fin}_i \to \mathsf{Fin}_j) \to \mathcal{U}$$
$$\mathsf{isIncr}_{i,j}(f) :\equiv \Pi_{x,y:\mathsf{Fin}_i}(x <^i y) \to (f(x) <^j f(y)),$$

expressing that a function is strictly monotonously increasing. Let us further write $\Delta_+(i,j)$ for the type $\Sigma\,(f : \mathsf{Fin}_i \to \mathsf{Fin}_j)\,.\,\mathsf{isIncr}_{i,j}(f)$. We then have a composition operator $\circ : \Delta_+(h,i) \to \Delta_+(i,j) \to \Delta_+(h,j)$, defined separately on each of the two components. This is a representation of strictly increasing functions such that $\circ$ is strictly associative, as observed in [14]. Unsurprisingly, this is enough to make Δ_+ a *category* in the sense that we will define later (see Definition 5).[1]

In the following, we use variable names $\vec{X}$, $\vec{x}$ instead of X, x to indicate that we have an element of a nested Σ-type, i.e. a tuple. With Δ_+ at hand, we define truncated semi-simplicial types (SST) simultaneously with skeletons (SK) and the morphism part of skeletons (written $\mathsf{SK}^{\to}$). These have the following types:

$$\begin{aligned}
&\mathsf{SST} &&: \mathbb{N}^s \to \mathcal{U} && \text{— we write } \mathsf{SST}_k \text{ instead of } \mathsf{SST}(k);\\
&\mathsf{SK} &&: \Pi_{k:\mathbb{N}^s}\ \mathsf{SST}_k \to \mathbb{N}^s \to \mathcal{U} && \text{— we write } \mathsf{SK}^n_{k,\vec{X}} \text{ instead of } \mathsf{SK}(k,\vec{X},n);\\
&\mathsf{SK}^{\to} &&: \Pi_{k:\mathbb{N}^s}\Pi_{\vec{X}:\mathsf{SST}_k}\Pi_{m,n:\mathbb{N}^s}\Pi_{f:\Delta_+(m,n)}\mathsf{SK}^n_{k,\vec{X}} \to \mathsf{SK}^m_{k,\vec{X}} && \\
& && && \text{— we write } \mathsf{SK}^{\to}_{k,\vec{X}} \text{ instead of } \mathsf{SK}^{\to}(k,\vec{X},m,n).
\end{aligned}$$

These type families can be explained as follows:

1. SST_k is the type of $(k-1)$-truncated semi-simplicial types.
2. Assume we have a $(k-1)$-truncated semi-simplicial type $\vec{X}$, where k is smaller than another given number n.[2] $\vec{X}$ allows us to form the type $\mathsf{SK}^n_{k,\vec{X}}$. This is the type of "partial boundaries" of an $(n-1)$-truncated semi-simplicial type. Intuitively, it has n points, $\binom{n}{2}$ lines, …, and $\binom{n}{k}$ cells on level $(k-1)$.
3. We think of $\mathsf{SK}_{k,\vec{X}}$ as a "functor" from Δ_+ to $\mathcal{U}$. Its morphism component is given by $\mathsf{SK}^{\to}_{k,\vec{X}}$: for any $f : \Delta_+(m,n)$, we get a function $\mathsf{SK}^n_{k,\vec{X}} \to \mathsf{SK}^m_{k,\vec{X}}$ which simply "removes" those cells that appear in the partial boundary of an $(n-1)$-simplex, but not in the partial boundary of an $(m-1)$-simplex.

At the same time as we define SST, SK and $\mathsf{SK}^{\to}$, we prove the following strict functor law for all $k,l,m,n : \mathbb{N}^s$, $\vec{X} : \mathsf{SST}_k$, and $f : \Delta_+(l,m)$, $g : \Delta_+(m,n)$:

$$\alpha_k(\vec{X},f,g) : \mathsf{SK}^{\to}_{k,\vec{X}}g \circ \mathsf{SK}^{\to}_{k,\vec{X}}f \overset{s}{=} \mathsf{SK}^{\to}_{k,\vec{X}}(g \circ f).$$

We define all the components by induction on k as follows. In the base case, we set $\mathsf{SST}_0 :\equiv \mathbf{1}$; $\mathsf{SK}^n_{0,\star} :\equiv \mathbf{1}$; $\mathsf{SK}^{\to}_{0,\star}f :\equiv \mathsf{id}_{\mathbf{1}}$; and $\alpha_0(\star,f,g) :\equiv \mathsf{refl}^s$. In the successor case, we

[1] Note that, for technical reasons, we include the initial object Fin_0. This explains the shift by 1: we have defined $\Delta_+(i,j) :\equiv \Sigma\,(f : \mathsf{Fin}_i \to \mathsf{Fin}_j)\,.\,\mathsf{isIncr}_{i,j}(f)$ instead of $\Sigma\,(f : \mathsf{Fin}_{i+1} \to \mathsf{Fin}_{j+1})\,.\,\mathsf{isIncr}_{i,j}(f)$.

[2] The definition works for $k \geq n$, but $k < n$ is the case that is important for the intuition.

choose

$$\begin{aligned}
\mathsf{SST}_{k+1} &:\equiv \Sigma\left(\vec{X} : \mathsf{SST}_k\right).\, \mathsf{SK}^{k+1}_{k,\vec{X}} \to \mathcal{U} \\
\mathsf{SK}^{n}_{k+1,(\vec{X},Y)} &:\equiv \Sigma\left(\vec{x} : \mathsf{SK}^{n}_{k,\vec{X}}\right).\, \Pi_{f:\Delta_+(k+1,n)} Y\left(\mathsf{SK}^{\to}_{k,\vec{X}}(f,\vec{x})\right) \\
\mathsf{SK}^{\to}_{k+1,(\vec{X},Y)}(f,(\vec{x},h)) &:\equiv \left(\mathsf{SK}^{\to}_{k,\vec{X}}(f,\vec{x})\ ,\ \lambda g.\alpha_{k*}(h(f\circ g))\right)
\end{aligned}$$

Note that, in the last line, the type of the term $h(f \circ g)$ is $Y(\mathsf{SK}^{\to}_{k,\vec{X}}(g \circ f, \vec{x}))$. However, what we need at that point is an element of the type $Y(\mathsf{SK}^{\to}_{k,\vec{X}}(g, \mathsf{SK}^{\to}_{k,\vec{X}}(f,\vec{x})))$. This is why we transport along the proof $\alpha_k(\vec{X}, f, g)$, abbreviated to α_k, which shows that the two types are strictly equal.

We omit the term for α_{k+1} as it is not insightful to write it down explicitly. It is constructed as follows. First, we note that we need a (strict) equality between pairs; the first components are (strictly) equal by α_k. When one tries to prove that the second components are (strictly) equal, one quickly realizes that what is needed is coherence for the family of strict equalities α_k: The composition $\mathsf{SK}^{\to}_{k,\vec{X}} g \circ \mathsf{SK}^{\to}_{k,\vec{X}} f \circ \mathsf{SK}^{\to}_{k,\vec{X}} e$ can be shown to be (strictly) equal to $\mathsf{SK}^{\to}_{k,\vec{X}}(g \circ f \circ e)$ in two ways, and we need that both ways are strictly equal. Of course, this follows from the fact that we have axiom K for our strict equality. We have verified this construction in Agda (see the remark below).

▶ Remark. We and many others have attempted to formalize semi-simplicial types in homotopy type theory with exactly the outlined strategy, replacing the strict law α by the usual equality type. This works in the same way until the point where we need that α_k is coherent, which is automatic in our case. Intuitively, α_k *is* coherent, and it is easy to get trapped into thinking that this coherence can just be shown simultaneously with the other four components. However, if one does this, one notices that one needs an additional coherence level for α_{k-1}, and it continues like this. Morally, all these coherences should hold, and it is very likely that we would actually be able to prove them inductively if only we were able to write them down. Unfortunately, writing them down is a problem that is very similar to defining semi-simplicial types itself. From this point of view, what the 2-level theory gives us is the possibility to prove a certain equality and *all* its higher coherences at the same time.

▶ Remark. We have now defined the n-truncated semi-simplicial types $\mathsf{SST} : \mathbb{N}_s \to \mathcal{U}$, so we may ask whether we can define a (non-truncated) type of semi-simplicial types $\mathsf{SS} : \mathcal{U}$. If we work in the strong 2-level theory (where $\mathbb{N}_s$ and $\mathbb{N}$ coincide), we can consider the homotopy limit $\mathsf{SS} : \mathcal{U}$, defined as

$$\mathsf{SS} :\equiv \Sigma\,(f : \Pi_{n:\mathbb{N}} \mathsf{SST}_n).\, \Pi_{i:\mathbb{N}} \mathsf{fst}(f(i+1)) = f(i)$$

Then, SS is indeed a (fibrant) type that encodes (Reedy fibrant) functors $\Delta_+^{\mathrm{op}} \to \mathcal{U}$.

▶ Remark. Our Agda formalisation[3] takes place within the fibrant theory. The contribution of the strict equality is completely encapsulated in a single lemma that we postulate without a formal proof. Unfortunately, simulating our 2-level system completely in Agda, although possible in principle, would be extremely cumbersome because of the need to keep track of type fibrancy manually.

[3] `https://github.com/nicolaikraus/HoTT-Agda/blob/master/nicolai/SemiSimp/SStypes.agda`

4 Reedy Fibrant Diagrams Over Inverse Categories

In Section 3, we have defined Reedy fibrant truncated semi-simplicial types using our 2-level theory. We have stayed in the fibrant theory (HoTT) as much as we could, and only used the strict theory to prove a crucially needed coherence. In this section we want to demonstrate that the 2-level theory is even more powerful if we give up this strategy of only working in the fibrant fragment whenever possible. The point is that we can derive results about HoTT without staying inside HoTT, analogous to how one can get results that respect homotopy equivalence even when certain constructions are performed on concrete spaces that only represent homotopy types.

What we claim is that, in a proof assistant implementing a 2-level type theory, we could formalize many constructions that are presented meta-theoretically in the current literature. In the current section, we will show that Reedy fibrant diagrams $I \to \mathcal{U}$ have limits in $\mathcal{U}$ if I is a finite inverse category. This is an internalised version of results discussed by Shulman [20]. Of course it generalizes the construction in Section 3, although not "literally": the truncated semi-simplicial types that we get here will look different from those constructed above.

4.1 Essentially Fibrant Pretypes and Strict Fibrations

As a preparation for our "more abstract" sample applications of the 2-level theory, we remark that it is often not necessary to know that a pretype $A : \mathcal{U}^s$ is a fibrant type. Instead, it is usually sufficient to have a fibrant type $B : \mathcal{U}$ and a strict isomorphism $A \simeq^s B$. If this is the case, we say that A is *essentially fibrant*. Clearly, every fibrant type is also an essentially fibrant pretype.

In Section 3, we have made heavy use of the fibrant finite types Fin_n (for $n : \mathbb{N}$). In a strong 2-level theory, this type coincides with the strict pretype $\mathsf{Fin}^s_n : \mathcal{U}^s$ (for $n : \mathbb{N}^s$), but this is not in general the case. We say that some pretype I is *essentially finite* if we have a number $n : \mathbb{N}^s$ and a strict isomorphism $I \simeq^s \mathsf{Fin}^s_n$.

▶ **Lemma 3.** *Let I be essentially finite and $X : I \to \mathcal{U}$ be a family of fibrant types. Then, $\Pi_{i:I} X(i)$ is essentially fibrant.*

Proof. Essential finiteness gives us a cardinality n on which we do induction. If n is $\mathbf{0}^s$, then $\Pi_{i:I} X(i)$ is strictly isomorphic to the unit type. Otherwise, we have an essentially finite I' such that $f : \mathbf{1} +^s I' \simeq^s I$, and $\Pi_{i:I} X(i)$ is strictly isomorphic to $X(f(\mathsf{inl}\ \star)) \times \Pi_{i:I'} X(f(\mathsf{inr}\ i))$, which is essentially finite by the induction hypothesis. ◀

Similar to essential fibrancy, we have the following definition:

▶ **Definition 4** (strict fibration)**.** Let $p : E \to B$ be a function (with $E, B : \mathcal{U}^s$). We say that p is a *strict fibration* if we have a family $F : B \to \mathcal{U}$ such that the fibre of p over any $b : B$ is strictly isomorphic to $F(b)$, that is, $\Pi_{b:B} \left(F(b) \simeq^s \Sigma\,(e : E)\,.\,p(e) \stackrel{s}{=} b \right)$.

From now on, we will drop the attribute *strict* and simply talk about *fibrations*. Any fibrant type family $F : B \to \mathcal{U}$ gives rise to a fibration $p : E \to B$, as it is easy to see that the first projection $(\Sigma_B F) \to B$ satisfies the given condition. Indeed, any strict fibration is isomorphic over B to a strict fibration of this form. This often allows us to assume that a given fibration has the form of a projection.

4.2 Strict Categories

We define categories in much the same way as the precategories are defined in [21], except that we use strict equality to express the laws. Since strict equality does not suffer from coherence issues, this notion of category is well-behaved. It can be applied to structures which do not have fibrant types of objects or morphisms.

▶ **Definition 5** (strict category)**.** A *strict category* $\mathcal{C}$ is given by: a pretype $|\mathcal{C}| : \mathcal{U}^s$ of *objects*; for all pairs $x, y : |\mathcal{C}|$, a pretype $C(x, y) : \mathcal{U}^s$ of *arrows* or *morphisms*; an *identity* arrow $\mathsf{id} : \mathcal{C}(x, x)$ for every object x; and a *composition* function $\circ : \mathcal{C}(y, z) \to \mathcal{C}(x, y) \to \mathcal{C}(x, z)$ for all objects x, y, z. The usual categorical laws are required to hold strictly, that is, we have strict equalities $f \circ \mathsf{id} \stackrel{s}{=} f$ and $\mathsf{id} \circ f \stackrel{s}{=} f$, as well as $h \circ (g \circ f) \stackrel{s}{=} (h \circ g) \circ f)$.

We say that a category is *essentially finite* if the pretype of objects $|\mathcal{C}|$ is essentially finite (no condition is put on the arrows).

The usual theory of categories can be reproduced in the context of strict categories. We leave it to the reader to define appropriate notions of *functor*, *natural transformation*, *limits*, *adjunctions*, and so on.

From now on, we will refer to strict categories simply as *categories*. If $\mathcal{C}$ is a category, we will often abuse notation and use $\mathcal{C}$ itself to denote its type of objects.

Another important notion is the following:

▶ **Definition 6** (reduced coslice)**.** Given a category $\mathcal{C}$ and an object $x : \mathcal{C}$, the *reduced coslice* $x /\!\!/ \mathcal{C}$ is the full subcategory of non-identity arrows in the coslice category $x/\mathcal{C}$. A concrete definition is the following. The objects of $x /\!\!/ \mathcal{C}$ are triples of an $y : |\mathcal{C}|$, a morphism $f : \mathcal{C}(x, y)$, and a proof $\neg \left(p_*(f) \stackrel{s}{=} \mathsf{id}\right)$, for all $p : x \stackrel{s}{=} y$, where p_* denotes the transport function $\mathcal{C}(x, y) \to \mathcal{C}(y, y)$. Morphisms between (y, f, s) and (y', f', s') are elements $h : \mathcal{C}(y, y')$ such that $h \circ f \stackrel{s}{=} f'$ in $\mathcal{C}$.

Note that we have a "forgetful functor" $\mathsf{forget} : x /\!\!/ \mathcal{C} \to \mathcal{C}$, given by the first projection on objects as well as on morphisms.

4.3 Inverse Categories

Classically, *inverse categories* are categories which do not contain an infinite sequence of nonidentity arrows (see [20]). We restrict ourselves to those which have *height* at most ω, and where a *rank function* is given explicitly. First, consider the category $\mathbb{N}^{s\mathrm{op}}$ which has $n : \mathbb{N}^s$ as objects, and $\mathbb{N}^{s\mathrm{op}}(n, m) :\equiv n >^s m$ (the function $>^s : \mathbb{N}^s \to \mathbb{N}^s \to \mathcal{U}^s$ is defined in the canonical way). Then, we define:

▶ **Definition 7** (inverse category)**.** We say that a category $\mathcal{C}$ is an *inverse category* if there is a functor $\varphi : \mathcal{C} \to \mathbb{N}^{s\mathrm{op}}$ which reflects identities; i.e. if we have $f : \mathcal{C}(x, y)$ and $\varphi_x \stackrel{s}{=} \varphi_y$, then we also have $p : x \stackrel{s}{=} y$ and $p_*(f) \stackrel{s}{=} \mathsf{id}$.

4.4 Reedy Fibrant Limits

Much of what is known about the category of sets in classical category theory can be extended to the category of pretypes in a given universe. For example, the following result translates rather directly:

▶ **Lemma 8.** *The universe $\mathcal{U}^s$, viewed as a category in the canonical sense, has all small limits.*

Proof. Let $\mathcal{C}$ be a category with $|\mathcal{C}| : \mathcal{U}^s$ and $\mathcal{C}(x,y) : \mathcal{U}^s$ (for all x, y). Let $X : \mathcal{C} \to \mathcal{U}^s$ be a functor. We define L to be the pretype of natural transformations $\mathbf{1} \to X$, where $\mathbf{1} : \mathcal{C} \to \mathbf{Type}$ is the constant functor on $\mathbf{1}$. Clearly, $L : \mathcal{U}^s$, and a routine verification shows that L satisfies the universal property of the limit of X. ◀

Unfortunately, the category $\mathcal{U}$ of fibrant types is not as well behaved. Even pullbacks of fibrant types are not fibrant in general (but see Lemma 9). If we have a functor $X : \mathcal{C} \to \mathcal{U}$, we can always regard it as a functor $X : \mathcal{C} \to \mathcal{U}^s$, where it does have a limit. If this limit happens to be essentially fibrant, we say that X has a *fibrant limit*. Clearly, this limit will then be a limit of the original diagram $C \to \mathcal{U}$ (note that $\mathcal{U}$ is a full subcategory of $\mathcal{U}^s$).

▶ **Lemma 9.** *The pullback of a fibration $E \to B$ along any function $f : A \to B$ is a fibration.*

Proof. We can assume that E is of the form $\Sigma\,(b : B)\,.\,C(b)$ and p is the first projection. Clearly, the first projection of $\Sigma\,(a : A)\,.\,C(f(a))$ satisfies the universal property of the pullback. ◀

Lemma 9 makes it possible to construct fibrant limits of certain "well-behaved" functors from inverse categories. The so-called *matching objects* play an important role.

▶ **Definition 10** (matching object; see [20, Chp. 11])**.** Let $\mathcal{C}$ be an inverse category, and $X : \mathcal{C} \to \mathcal{U}$ a functor. For any $z : \mathcal{C}$, we define the *matching object* M_z^X to be the (not necessarily fibrant) limit of the composition $z /\!\!/ \mathcal{C} \xrightarrow{\mathsf{forget}} \mathcal{C} \xrightarrow{X} \mathcal{U} \subset \mathcal{U}^s$.

▶ **Definition 11** (Reedy fibrant diagram; see [20, Def. 11.3])**.** Let $\mathcal{C}$ be an inverse category and $X : \mathcal{C} \to \mathcal{U}$ be a functor. We say that X is *Reedy fibrant* if, for all $z : \mathcal{C}$, the canonical map $X_z \to M_z^X$ is a fibration.

Using this definition, we can make precise the claim that we can construct fibrant limits of certain well-behaved diagrams:

▶ **Theorem 12** (see [20, Lemma 11.8])**.** *Let $\mathcal{C}$ be an essentially finite inverse category. Then, every Reedy fibrant $X : \mathcal{C} \to \mathcal{U}$ has a fibrant limit.*

Proof. By induction on the cardinality of $\mathcal{C}$. In the zero case, the limit is the unit type.

Otherwise, let us consider the rank functor $\varphi : \mathcal{C} \to \mathbb{N}^{s^{\mathrm{op}}}$. We choose an object $z : \mathcal{C}$ such that φ_z is maximal; this is possible (constructively) since $\mathcal{C}$ is assumed to be essentially finite. Let us call $\mathcal{C}'$ the category that we get if we remove z from $\mathcal{C}$; that is, we set $|\mathcal{C}'| :\equiv \Sigma\,(x : |\mathcal{C}|)\,.\,\neg(x \overset{s}{=} z)$. Clearly, $\mathcal{C}'$ is still essentially finite and inverse.

Let $X : \mathcal{C} \to \mathcal{U}$ be Reedy fibrant. We can write down the limit of X explicitly as

$$\Sigma\left(c : \Pi_{y:|\mathcal{C}|} X_y\right)\,.\,\Pi_{y,y':|\mathcal{C}|}\Pi_{f:\mathcal{C}(y,y')} Xf(c_y) \overset{s}{=} c_{y'}. \tag{1}$$

Using that z has no incoming non-identity arrows, this pretype is strictly isomorphic to

$$\begin{aligned}\Sigma\,(c_z : X_z)\,.\,\Sigma\left(c : \Pi_{y:|\mathcal{C}'|} X_y\right)\,.\\ \left(\Pi_{y:|\mathcal{C}'|}\Pi_{f:\mathcal{C}(z,y)} Xf(c_z) \overset{s}{=} c_y\right) \times \left(\Pi_{y,y':|\mathcal{C}'|}\Pi_{f:\mathcal{C}(y,y')} Xf(c_y) \overset{s}{=} c_{y'}\right).\end{aligned} \tag{2}$$

Let us write L for the limit of X restricted to $\mathcal{C}'$, and let us further write p for the canonical map $p : L \to M_z^X$. Further, we write q for the map $X_z \to M_z^X$. Then, (2) is strictly isomorphic to

$$\Sigma\,(c_z : X_z)\,.\,\Sigma\,(d : L)\,.\,p(d) \overset{s}{=} q(c_z). \tag{3}$$

This is the pullback of the span $L \xrightarrow{p} M_z^X \xleftarrow{q} X_z$. By Reedy fibrancy of X, the map q is a fibration. Thus, by Lemma 9, the map from (3) to L is a fibration.

By the induction hypothesis, L is essentially fibrant. This implies that (3) is essentially fibrant, as it is the domain of a fibration whose codomain is essentially fibrant. ◀

If $\mathcal{C}$ is an inverse category, we will denote by $\mathcal{C}^{<n}$ the full subcategory of $\mathcal{C}$ consisting of all those objects of rank less than n. Correspondingly, for a given diagram X over $\mathcal{C}$, we will denote by $X|n$ the restriction of X to $\mathcal{C}^{<n}$.

4.5 Fibrant Limits and Semi-Simplicial Types

If X is a Reedy fibrant diagram over $\mathcal{C} :\equiv \left(\Delta_+^{\mathrm{op}}\right)^{<n}$, we can restrict X to $n /\!\!/ \mathcal{C}$, then take the limit of the corresponding functor. With a slight abuse of notation, we will denote such limit by M_n^X, even though X is not defined at n.

Note that a diagram X over $\left(\Delta_+^{\mathrm{op}}\right)^{<n+1}$ is Reedy fibrant if and only if its restriction to $\left(\Delta_+^{\mathrm{op}}\right)^{<n}$ is Reedy fibrant and the map $X_n \to M_n^X$ is a fibration. Hence, to give a Reedy fibrant diagram over $\left(\Delta_+^{\mathrm{op}}\right)^{<n+1}$ is the same as to give a Reedy fibrant diagram X over $\left(\Delta_+^{\mathrm{op}}\right)^{<n}$, together with a fibration Y over M_n^X. We will refer to this extended diagram as $\langle X, Y\rangle$. By mutual induction on the natural number n, we can define a type SST_n, and a function SK_n from SST_n to diagrams over $\left(\Delta_+^{\mathrm{op}}\right)^{<n}$. We start with with $\mathsf{SST}_0 :\equiv \mathbf{1}$ and $\mathsf{SK}_0(\star)$ set to the trivial diagram over $\left(\Delta_+^{\mathrm{op}}\right)^{<0}$. Then, we set

$$\mathsf{SST}_{n+1} :\equiv \Sigma\,(X : \mathsf{SST}_n)\,.\,(M_n^{\mathsf{SK}_n X} \to \mathcal{U}) \qquad \text{and} \qquad \mathsf{SK}_{n+1}(X, Y) :\equiv \langle X, Y\rangle.$$

Above, we write M_n^A to mean the type given by Theorem 12 which is strictly isomorphic to the matching object of A at n (which would otherwise only be a pretype).

This gives us a succinct alternative to the construction of Section 3, where most of the hard work is encapsulated in the use of Theorem 12.

5 Conclusions and Further Work

In the previous two sections, we have demonstrated how our 2-level theories can be used in two ways. First, our framework offers reasonable, easily justifiable ways of extending homotopy type theory. Second, we can internalise results about homotopy type theory that, before, could only be stated meta-theoretically. In a suitable proof assistant which implements a 2-level theory, we could formalize many constructions that can at the moment only be done on paper. Our current article offers a demonstration of this possibility: we have shown how some of the constructions about fibrant limits and diagrams can be internalised. From here, we could go into several directions. We could, for example, internalise Shulman's result that diagrams over a model of type theory form again a model, preserving univalence [20]. Of course, for such an internalisation, we need to be careful to formulate all definitions and results constructively.

A more modest but (as we believe) worthwhile next goal is the construction of fibrant replacements. With this, we can internalise the proof that any type carries the structure of an ∞-groupoid (a Kan semi-simplicial type), as it is given in [13, Remark and Corollary 16]. To do this, we would first define an ∞-groupoid to be a Reedy fibrant semi-simplicial type $X : \Delta_+^{\mathrm{op}} \to \mathcal{U}$ such that every fibration from X_n to a horn is an equivalence (in the sense of homotopy type theory). We can then, for a type $A : \mathcal{U}$, consider the semi-simplicial type Eq_A, defined to be the Reedy fibrant replacement of the functor that is constantly A.

It is shown in [13] that Eq_A is an ∞-groupoid in our sense, and the argument can easily be internalised. This construction is in fact not difficult and has in the current paper been omitted solely for reasons of space.

Our next significant project, supported by the 2-level theory, is the development of $(\infty, 1)$-category theory. By an $(\infty, 1)$-category, which could also be called a *Segal type*, we mean a Reedy fibrant semi-simplicial type X for which the usual "Segal maps" $X_n \to X_1 \times_{X_0} \ldots \times_{X_0} X_1$ are equivalences. It is likely that it is necessary to add degeneracies, and we expect that this can be done in the way presented by Harpaz [7].

We believe that it is important to develop a theory of $(\infty, 1)$-categories type-theoretically, because the universe itself should be an $(\infty, 1)$-category; we expect that many infinite coherence problems become approachable if we can set up some basic infrastructure, so that towers of coherences could be formulated and handled in a clean way.

The most important application that we currently have in mind is the specification of *higher inductive types* (HITs). Although HITs are used frequently in the literature on homotopy type theory, we do not have a general syntactical specification yet. The approach to define a general syntactical framework of HITs that is used in [2] seems to be promising, but suffers from the issue that an unmanageable number of coherences needs to be handled manually. We expect and hope that this can be resolved with the framework of $(\infty, 1)$-categories that we plan to develop.

References

1. Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science (MSCS)*, pages 1–30, Jan 2015. doi:10.1017/S0960129514000486.
2. Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, and Fredrik Nordvall Forsberg. Towards a theory of higher inductive types. Presentation at TYPES'15, Tallinn, Estonia, 20 May 2015.
3. Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Infinite structures in type theory: Problems and approaches. Presented at TYPES'15, Tallinn, Estonia, 20 May 2015.
4. Andrej Bauer, Gaëtan Gilbert, Philipp Haselwarter, Matija Pretnar, and Chris Stone. Andromeda. Implementation of a type theory with equality reflection, ongoing project. URL: http://andromedans.github.io/andromeda/.
5. Paolo Capriotti. *Models of Type Theory with Strict Equality*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2016. In preparation.
6. Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs (TYPES)*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer-Verlag, 1995. doi:10.1007/3-540-61780-9_66.
7. Yonatan Harpaz. Quasi-unital ∞–categories. *Algebraic & Geometric Topology*, 15(4):2303–2381, 2015.
8. Hugo Herbelin. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science (MSCS)*, pages 1–16, Mar 2015. doi:10.1017/S0960129514000528.
9. Martin Hofmann. Conservativity of equality reflection over intensional type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs (TYPES)*, volume 1158 of *Lecture Notes in Computer Science*, pages 153–164. Springer-Verlag, 1995. doi:10.1007/3-540-61780-9_68.
10. Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.

11 Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Venice Festschrift*, pages 83–111. Oxford University Press, 1996.

12 Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations. *ArXiv e-prints*, Nov 2012.

13 Nicolai Kraus. The general universal property of the propositional truncation. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *Types for Proofs and Programs (TYPES)*, volume 39 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111–145, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2014.111.

14 Nicolai Kraus. *Truncation Levels in Homotopy Type Theory*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2015.

15 Peter LeFanu Lumsdaine. *Higher Categories from Type Theories*. PhD thesis, Carnegie Mellon University, 2010.

16 Jacob Lurie. *Higher Topos Theory*. Annals of Mathematics Studies. Princeton University Press, Princeton, 2009.

17 Maria Emilia Maietti. A minimalist two-level foundation for constructive mathematics. *Annals of Pure and Applied Logic*, 160(3):319–354, 2009. Computation and Logic in the Real World: CiE 2007. doi:10.1016/j.apal.2009.01.006.

18 Fedor Part and Zhaohui Luo. Semi-simplicial types in logic-enriched homotopy type theory. *CoRR*, abs/1506.04998, 2015. URL: http://arxiv.org/abs/1506.04998.

19 Michael Shulman. Homotopy type theory should eat itself (but so far, it's too big to swallow). Blog post, homotopytypetheory.org/2014/03/03/hott-should-eat-itself, 3 Mar 2014.

20 Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, pages 1–75, Jan 2015. doi:10.1017/S0960129514000565.

21 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. URL: http://homotopytypetheory.org/book/.

22 Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011. doi:10.1112/plms/pdq026.

23 Vladimir Voevodsky. A simple type system with two identity types, 2013. Unpublished note.